Towards the Reverse-Engineering of the CaveTable

Sohaïb Ouzineb

Internship Report

Towards the Reverse-Engineering of the CaveTable

Author:
Sohaïb Ouzineb
sohaib.ouzineb@telecom-paris.fr

Supervisor:
Dr. Léo Perrin
leo.perrin@inria.fr

A report submitted during a Summer internship in the

Project Team SECRET
INRIA, the French Research Institute for Digital Sciences

August 30, 2019

Abstract

Towards the Reverse-Engineering of the CaveTable

by Sohaïb Ouzineb

This report focuses on the S-Box used in CMEA . The purpose is to find the generation process of this S-Box. Such knowledge is important since a weak design process usually results in vulnerability against attacks. Many of them have already been published against CMEA. The attackers used the statistical bias in the S-Box to develop their attack. We want to know what caused such a statistical bias, so that this method may be avoided later on. In order to find a structure in this S-Box, we first recall the high level structure that has already been found before. We then look at several properties, such as the linear and differential properties and the hardware related properties, and finally, the relations with other cryptographic algorithms. Our results show that the TU structure is likely just a consequence of the S-Box being generated from thirty-two 4×4 permutations being concatenated. The properties of the tables components involved in the structure are all consistent with those of pseudo-random permutations. Hence, the S-Box might just be constructed using thirty-two pseudo-randomly generated 4×4 permutations.

Contents

1 Introduction

Cryptographic algorithms usually use a substitution box called S-Box. Such algorithms have to be non-linear, else they would be very weak as a Gaussian elimination would break it. The S-Box is often the only component that is non-linear. Their design criteria differ from one another. However, the main goal is to get them as resistant to attacks as possible. When a new S-Box is used, there should be some calculus to indicate that the S-Box is strong. There should also be some indication to show how the S-Box was generated. At first glance, it may seem like explaining the design criteria is like revealing confidential information. However, hiding the design criteria may results in distrust from clients because we are not sure that this S-Box is sufficiently resistant. The designer may for example have hidden a backdoor (a hidden mathematical property to simplify attacks if the attacker knows it). The S-Box may also not be strong enough and the designers could hide this fact by just giving the table.
CMEA is a byte oriented block cipher that is used in the North American digital cellular systems. It is one of the four cryptographic primitives described by the Telecommunications Industry Association in 1992 presented in [B92], and has been updated in 1995 [Qui09]. It is used to encrypt the digits dialed by the user. CMEA consists of 3 layers, the first one is non linear and the last one is the inverse of the first one. The second layer is linear and consists in XORing the right part of the block into the left one. The first and last layers use a T-Box that is derived from an S-Box. The generation process of the T-Box is found in Section 6.1. The goal here is to find the generation process of CMEA S-Box S, which is also named the CaveTable. The look-up-table of this S-Box is found in Table 1.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	d9	23	5f	e6	ca	68	97	b0	7b	f2	0c	34	11	a5	8d	4e
1.	0a	46	77	8d	10	9f	5e	62	f1	34	ec	a5	c9	b3	d8	2b
2.	59	47	e3	d2	ff	ae	64	ca	15	8b	7d	38	21	bc	96	00
3.	49	56	23	15	97	e4	cb	6f	f2	70	3c	88	ba	d1	0d	ae
4.	e2	38	ba	44	9f	83	5d	1c	de	ab	c7	65	f1	76	09	20
5.	86	bd	0a	f1	3c	a7	29	93	cb	45	5f	e8	10	74	62	de
6.	b8	77	80	d1	12	26	ac	6d	e9	cf	f3	54	3a	0b	95	4e
7.	b1	30	a4	96	f8	57	49	8e	05	1f	62	7c	c3	2b	da	ed
8.	bb	86	0d	7a	97	13	6c	4e	51	30	e5	f2	2f	d8	c4	a9
9.	91	76	f0	17	43	38	29	84	a2	db	ef	65	5e	ca	0d	bc
A.	e7	fa	d8	81	6f	00	14	42	25	7c	5d	c9	9e	b6	33	ab
B.	5a	6f	9b	d9	fe	71	44	c5	37	a2	88	2d	00	b6	13	ec
C.	4e	96	a8	5a	b5	d7	c3	8d	3f	f2	ec	04	60	71	1b	29
D.	04	79	e3	c7	1b	66	81	4a	25	9d	dc	5f	3e	b0	f8	a2
E.	91	34	f6	5c	67	89	73	05	22	aa	cb	ee	bf	18	d0	4d
F.	f5	36	ae	01	2f	94	c3	49	8b	bd	58	12	e0	77	6c	da

TABLE 1: CaveTable

2 Mathematical Background

2.1 S-Box

We define an $n \times m$ S-Box S as a function

$$S : \mathbb{F}_2^n \to \mathbb{F}_2^m \ .$$

For convenience, we may prefer using the hexadecimal notation. Hence, an 8×8 S-Box may be seen as :

$$S : \mathbb{F}_2^4 \times \mathbb{F}_2^4 \to \mathbb{F}_2^4 \times \mathbb{F}_2^4 \ ,$$

where each symbol is seen as a hexadecimal symbol.

2.2 Mathematical Operations

We note the XOR function as $\oplus$. The '+' symbol refers to modular addition.

2.3 Look-up Table

There are many ways to represent an S-Box. In this article, we use the look-up table. The look-up table is a table that is read as follows:

- the input is made of the concatenation of the row number and the column number (using the hexadecimal notation),
- the output is the content of the corresponding position.

For example, when we look at Table 1, more precisely the content at the first row and second column, we get $S(12) = 77$ (using the hexadecimal notation).

2.4 Difference Distribution Table (DDT)

Let S be an $n \times m$ S-Box. The Difference Distribution Table (DDT) is a $2^n \times 2^m$ table such that for all $(a, b) \in F_2^n \times F_2^m$:

$$\mathrm{DDT}[a, b] = \#\left\{x \in F_2^n \ , S(x \oplus a) \oplus S(x) = b\right\} \ .$$

We also the define the differential spectrum of an S-Box as:

$$\left\{(k, \sigma), k = \#\left\{(a, b), \mathrm{DDT}[a, b] = \sigma\right\}\right\} \ .$$

Having low DDT coefficients is crucial so that the S-Box may resist differential attacks. The reader may refer to [BS91] for more information. The relevance of such a table for our purpose is explained in Section 5.3.

2.5 Linear Approximation Table (LAT)

We first define the inner product in $\mathbb{F}_2^n$:

$$x \cdot y = \bigoplus_{i=1}^{n} x_i y_i \ ,$$

where x_i is the i-th bit of the binary representation of x. Let S be an $n \times m$ S-Box. We define the Linear Approximation Table (LAT) as $2^n \times 2^m$ table such that for all $(a, b) \in F_2^n \times F_2^m$:

$$\mathrm{LAT}[a, b] = \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x + b \cdot S(x)} \ .$$

The '+' symbol denotes the XOR operation here. We also the define the linear spectrum of an S-Box as:

$$\left\{(k, \sigma), k = \#\left\{(a, b), |\mathrm{LAT}[a, b]| = \sigma\right\}\right\} \ .$$

The reader may not only refer to [MY93], but also to [TCG92].

2.6 Affine Equivalence

A linear mapping is a function L such that

$$L : \mathbb{F}_2^n \to \mathbb{F}_2^n \ ,$$

and for all x, y in $\mathbb{F}_2^n$,

$$L(x \oplus y) = L(x) \oplus L(y) \ .$$

Hence, an affine mapping is a function A, such that $A(x) = L(x) + c$ for a certain c in $\mathbb{F}_2^n$.

Let S_1 and S_2 be two $n \times n$ permutations. S_1 and S_2 are affine-equivalent if there exists two affine mappings A_1 and A_2 such that

$$S_2 = A_1 \circ S_1 \circ A_2 \ .$$

In this case, we note

$$S_1 \sim S_2 \ .$$

Such an equivalence preserves the differential and linear spectra.

2.7 Algebraic Normal Form (ANF)

Let S be an $n \times m$ S-Box. Let S_i be the i-th coordinate of S (using the binary representation). We can see S_i as a function of the input bits, $S_i = f(x_{n-1}, ..., x_1, x_0)$. We can then write f as XORs of ANDs. For example, we can have $S_2 = x_3x_1 \oplus x_1 \oplus 1$. This expression is called the algebraic normal form. We can generalize this to a component of S, a component of S being the sum of some of its coordinates. The algebraic degree of a component is the maximum number of input bits that we multiply together. For example, the algebraic degree of S_2 given above is 2.

3 High Level Structure

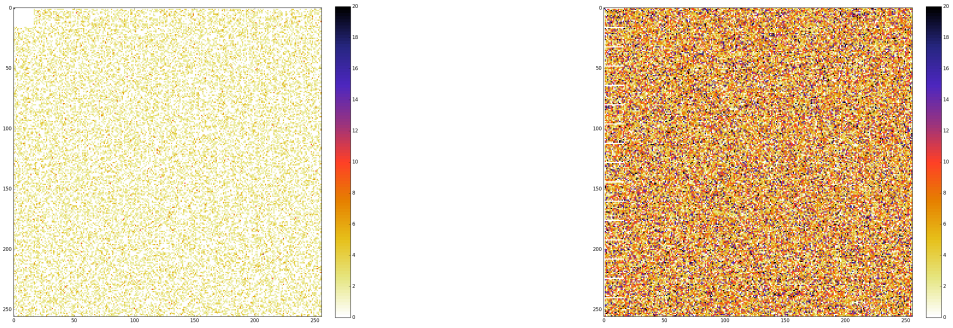
Before going into the details of the S-Box, we would like to first examine the S-Box as a whole to find a high level structure. In this part, I intend to summarize the main results that have been found in [Per17, Chaper 12].

We would like to compute the DDT to find interesting properties. However, the raw DDT may not be easy to analyse because it is too large. Hence, we use the Jackson-Pollock representation. The reader may find the sage [Dev17] function `save_pollock` in sBoxU [Per19] which is a way to show the DDT with varying levels of colors to show the intensity of the coefficients. Let us compute the Jackson-Pollock representation of both the DDT (Figure 1a) and the LAT (Figure 1b) of the CaveTable.

The white square at the top left of Figure 1a is equivalent to having a TU-structure, which means that we can write S as

$$S(x, y) = (T_x(y), U_{T_x(y)}(y)) \ ,$$

where the T_x are all 4×4 permutations. This structure is graphically represented in Figure 2.



(A) DDT CaveTable (B) LAT CaveTable

FIGURE 1: DDT and LAT of the CaveTable

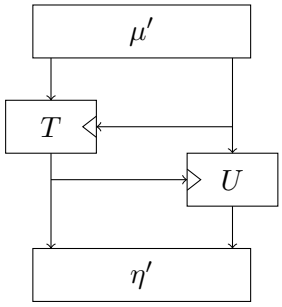


FIGURE 2: The TU-decomposition.

The white dents in Figure 1b show that if we compose S with $W : x||y \rightarrow y||x$, then we have a TU-Core composition (a TU composition without the affine mappings at the input or the output).

Here is the sage code snippet that allows the computation of the tables T and U:

```
def W(n):
   low = n & 0xf
   high=n>>4
   return (low<<4|high)

def get_tu_decomposition(S):
   T=[[0for i in range(16)] for j in range (16)]
   U=[[0for i in range(16)] for j in range (16)]
   for x in range(16):
      for y in range(16):
         T[x][y]=S[W(y<<4|x)] & 0xf
         U[T[x][y]][x]= S[W(y<<4|x)] >>4
   return (T,U)
(T,U)=get_tu_decomposition(S)
```

The tables T and U may be found respectively in Table 2 and Table 3.

The lines of T are permutations and the columns of U are too. We can represent this decomposition in multiple ways, which imply the definition of new tables to analyse. The first representation is the serial representation, given in Figure 3a. The second one is the parallel representation, given in Figure 3b.

The table U' of Figure 3b may hence be computed as follows : $U'_x(y) = U^t_x(T_x(y))$

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	9	3	f	6	a	8	7	0	b	2	c	4	1	5	d	e
1.	a	6	7	d	0	f	e	2	1	4	c	5	9	3	8	b
2.	9	7	3	2	f	e	4	a	5	b	d	8	1	c	6	0
3.	9	6	3	5	7	4	b	f	2	0	c	8	a	1	d	e
4.	2	8	a	4	f	3	d	c	e	b	7	5	1	6	9	0
5.	6	d	a	1	c	7	9	3	b	5	f	8	0	4	2	e
6.	8	7	0	1	2	6	c	d	9	f	3	4	a	b	5	e
7.	1	0	4	6	8	7	9	e	5	f	2	c	3	b	a	d
8.	b	6	d	a	7	3	c	e	1	0	5	2	f	8	4	9
9.	1	6	0	7	3	8	9	4	2	b	f	5	e	a	d	c
A.	7	a	8	1	f	0	4	2	5	c	d	9	e	6	3	b
B.	a	f	b	9	e	1	4	5	7	2	8	d	0	6	3	c
C.	e	6	8	a	5	7	3	d	f	2	c	4	0	1	b	9
D.	4	9	3	7	b	6	1	a	5	d	c	f	e	0	8	2
E.	1	4	6	c	7	9	3	5	2	a	b	e	f	8	0	d
F.	5	6	e	1	f	4	3	9	b	d	8	2	0	7	c	a

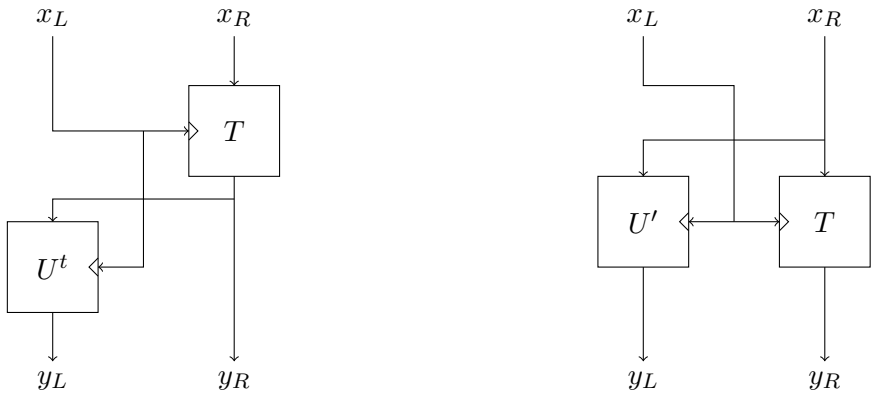
Table 2: T

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	b	1	0	7	2	1	8	3	3	f	0	0	6	b	d	e
1.	1	f	2	d	f	f	d	b	5	9	8	7	7	8	9	0
2.	f	6	d	f	e	6	1	6	f	a	4	a	f	a	2	1
3.	2	b	e	2	8	9	f	c	1	4	3	1	c	e	7	c
4.	3	3	6	e	4	7	5	a	c	8	1	4	0	0	3	9
5.	a	a	1	1	6	4	9	0	e	6	2	c	b	2	0	f
6.	e	4	9	5	7	8	2	9	8	7	b	b	9	6	f	3
7.	9	7	4	9	c	a	7	5	9	1	e	3	d	c	6	7
8.	6	d	3	8	3	e	b	f	d	3	d	8	a	f	1	5
9.	d	c	5	4	0	2	e	4	a	2	c	d	2	7	8	4
A.	c	0	c	b	b	0	3	d	7	c	f	5	5	4	a	d
B.	7	2	8	c	a	c	0	2	b	d	a	9	1	1	c	8
C.	0	e	b	3	1	3	a	7	6	b	7	e	e	d	5	6
D.	8	8	7	0	5	b	6	e	0	0	5	2	8	9	4	b
E.	4	5	a	a	d	d	4	8	4	5	9	f	4	3	e	a
F.	5	9	f	6	9	5	c	1	2	e	6	6	3	5	b	2

Table 3: U

4 Hardware Related Properties

S-Boxes are not just theoretical objects, they have to be physically implemented. CMEA had to be implemented on cellphones that had little computing power and memory storage. Most S-Boxes are not optimized for a hardware implementation. We would like to know if having an efficient hardware implementation was a design criteria for the CaveTable. The idea here is to analyse the hardware related properties of the S-Box in order to find if the latter is indeed optimized compared to a pseudo-random S-Box. There are many variables that we may look at. We first want to find about the implementation properties and then the algebraic properties.



(A) Serial representation. (B) Parallel representation.

FIGURE 3: First decompositions of CMEA SBox.

4.1 Gate Equivalent Complexity

In this part, we use LIGHTER presented in [JPST17] and PEIGEN presented in [BGLS19]. They are algorithms used to analyse some properties of a given S-Box. More precisely, they can compute the best hardware implementation with a given criteria, for example minimizing the propagation time. In this report, we will focus on the Gate Equivalent Complexity. The Gate Equivalent Complexity (GEC) of a certain S-Box S is the sum of the costs of all the logic gates that are used in the hardware implementation. Using LIGHTER, we may choose a set of logic gates with corresponding costs and then compute the GEC of an S-Box using the look-up table. We may also use the bit-slice representation in LIGHTER, but we use the look-up table here). The GEC of the permutations T_i are to be found in Table 4 .

Component	GEC (best found)	Number of implementations found
T_0	17.67	4
T_1	20.34	2
T_2	17.67	3
T_3	19.01	3
T_4	22.00	2
T_5	18.00	4
T_6	21.00	2
T_7	19.34	4
T_8	19.34	4
T_9	20.34	1
T_{10}	21.99	1
T_{11}	18.34	2
T_{12}	18.34	3
T_{13}	21.67	2
T_{14}	20.01	2
T_{15}	18.00	6

TABLE 4: GEC of the T_i

The best GEC found is 19.57 on average. It is slightly higher than that of a pseudo-random 4×4 permutation. The average best GEC found for a pseudo-random 4×4 permutation is 18.69. The code to compute the GEC of pseudo-random 4×4

permutations may be found in Appendix 1. The U_i^t have a best GEC of 18.63 on average, and 18.39 for the U_i'. As for the 8×8 S-Box, the CaveTable, the algorithm has run on a 192 GB RAM machine for 6 days and could not finish because of a lack in RAM memory. It is highly possible that the memory complexity of this problem is too high. Indeed, this algorithm uses a meet-in-the-middle method, which has an exponential complexity not only in time, but also in memory which happens to be the problem here. That being said, as the components of the generated tables (T, U, U', U^t) are not better than pseudo-random ones, we expect that it would be the same for the whole S-Box.

4.2 Algebraic Normal Form

Using a sage function, we can compute the algebraic degree of T components or of T as a whole. To do that, we use the function `algebraic_degree` that can be found in sBoxU [Per19].

```
from sboxU import *
for i in range(16):
   print(algebraic_degree(T[i]))
```

The components of T all have an algebraic degree equal to 3, which is consistent with that of a pseudo-random 4×4 permutation. To verify it, here is a small code snippet:

```
perm=range(16)
shuffle(perm)
print(algebraic_degree(perm))
Tlist=[]
for i in range(16):
    Tlist+=T[i]
print(algebraic_degree(Tlist))
```

If we compute the algebraic degree of T, we find 7. It is similar to what we get with an 8×4 S-Box made of pseudo-random 4×4 permutations. Again, here is a small code snippet to verify it:

```
pseudo_random=[]
for i in range(16):
     perm=range(16)
     shuffle(perm)
     pseudo_random+=perm
print(algebraic_degree(pseudo_random))
```

The algebraic degree of U is 7 and the same is true for U^t and U'. The raw ANF expression of the components is not exploitable as it is way too long and dense. Those of the $T[i + k_1] + k_0$ ('+' being the addition modulo 16) have the same properties.

5 Differential and Linear Properties

5.1 Affine Equivalence

We here want to look at the affine equivalence between the different components that we have (from U, U^t, T, U') so that we may express relations between them. The affine equivalence algorithm has a high cost in terms of algorithmic complexity. Hence, before starting the algorithm, we would prefer to first verify if both components have

the same DDT spectrum and the same LAT spectrum (in absolute value). To do that, the idea is to generate a dictionary whose keys correspond to a couple (DDT spectrum, LAT absolute spectrum) and whose values are the names of the components that have the same key. However, in a first part, as we do not have many components to analyse, we may then just want to classify them by their DDT spectrum. A sage code snippet to compute the corresponding dictionary may be found in Appendix 2.1.

The result is the following (keys are integers and correspond to a DDT spectrum, they are generated using the function `spectrum_to_int` available in Appendix 2.1, the lists correspond to the permutation names that have the same DDT spectrum):

```
17587605: ['Ui9'],
1006230: ['Ui0', 'Ti4'],
1201305: ['Utransposei10', 'Ui11', 'Ui14'],
17977755: ['Ui10', 'Utransposei12'],
51335580: ['Utransposei13'],
34754205: ['Utransposei0', 'Ti1', 'Utransposei1', 'Utransposei2', 'Ui3',
    'Ui4', 'Ti6', 'Ti13', 'Ti14', 'Ti15'],
68112030: ['Ui2', 'Ti5'],
51530655: ['Ui5', 'Utransposei14'],
4329720480: ['Ti7', 'Utransposei8'],
4313139105: ['Ti0', 'Ui12'],
51725730: ['Ti3', 'Ti9', 'Utransposei9'],
4329915555: ['Ui13'],
68502180: ['Ui15'],
101860005: ['Utransposei4', 'Ti12', 'Utransposei15'],
85278630: ['Ti2', 'Utransposei7'],
4396826280: ['Utransposei6'],
4330305705: ['Ui8'],
1099630197930: ['Utransposei5'],
17314692780: ['Utransposei3'],
34949280: ['Ti10'],
68307105: ['Ui1', 'Ui6', 'Ui7'],
4396436130: ['Ti8'],
4330110630: ['Ti11', 'Utransposei11']
```

For each key, we can see a list representing the components that have the same DDT spectrum. For example, $U_0^t, T_1, U_1^t, U_2^t, U_3, U_4, T_6, T_{13}, T_{14}, T_{15}$ have the same DDT spectrum. Using the function `affine_equivalence` that can be found in sBoxU [Per19], we get that the components cited earlier are not affine equivalent.

We now want to take a look at the components by adding a constant in both the input and the output, and then checking if some of them are affine equivalent. We hence look at the $T_i[x + k_1] + k_0$ that we note as $R(i, k_1, k_0)$, with '+' being the addition modulo 16. More precisely,

$$R(i, k_1, k_0) : \begin{cases} \mathbb{F}_2^4 & \to \mathbb{F}_2^4 \\ x & \mapsto T_i[x + k_1] + k_0 \ . \end{cases}$$

To this end, we classify them by (DDT spectrum, LAT absolute spectrum). A sage code snippet to compute the corresponding dictionary may be found in Appendix 2.2.

The dictionary is very long, there is no utility in showing it there. We now want to browse it by key and, for each key, browse the values list and check if some components are affine equivalent. The sage script may be found in Appendix 2.3.

We do not see any interesting pattern, except those corresponding to the following properties.

Lemma 1. *Adding 4 to a 4×4 permutation preserves the affine equivalence.*

Proof. Let S be 4×4 permutation, Let $E = S + 4$ (modulo 16), we then have that the coordinates of E are:

$$E = S_3 \oplus S_2, S_1 \oplus 1, S_1, S_0 \ .$$

Let

$$A : x_3, x_2, x_1, x_0 \rightarrow x_3 \oplus x_2, x_2, x_1, x_0 \ .$$

A is a linear mapping. We then have that

$$E = A(S \oplus 0100) \ .$$

S and E are then affine equivalent. □

Corollary 1. *Adding multiples 4 to k_0 in R preserves the affine equivalence.*

$$R(i, k_1, k_0) \sim R(j, k_1^j, k_0^j) \Leftrightarrow R(i, k_1, k_0) \sim R(j, k_1^j, k_0^j + 4\lambda) \quad (\lambda \in \mathbb{Z}) \ .$$

Proof. Using lemma 1 and transitivity we get that:

$$R(i, k_1, k_0) \sim R(j, k_1^j, k_0^j) \sim R(j, k_1^j, k_0^j + 4) \sim R(j, k_1^j, k_0^j + 4\lambda) \quad (\lambda \in \mathbb{Z}) \ .$$

□

Corollary 2. *Adding multiples 4 to either k_0 or k_1 in R preserves the affine equivalence.*

$$R(i, k_1, k_0) \sim R(j, k_1^j, k_0^j) \Leftrightarrow R(i, k_1, k_0) \sim R(j, k_1^j + 4\lambda', k_0^j + 4\lambda) \quad (\lambda, \lambda' \in \mathbb{Z}) \ .$$

Proof. We apply the reasoning of the previous proof for both k_1 and k_0. □

5.2 Branching Number

The S-Box decomposition is kind of similar to the DES one. Indeed, in the DES, 2 bits are used to choose a certain permutation, which is then applied to the 4 remaining bits. Knowing that the DES has a branching number equal to 3, which is unusual for an S-Box except if it is an explicit design criteria, we may want to compute the branching number of the CaveTable.

The branching number of an S-Box S is (hw being the Hamming weight):

$$\min_{(a,b)\neq(0,0)} \{\mathrm{hw}(a) + \mathrm{hw}(b), \mathrm{DDT}[a, b] > 0\} \ .$$

Here is a little sage code snippet to compute the branching number:

```
def popcount_py(x): #hamming weight
   return bin(x).count("1")

def branching_number(S):
   min=8
   DDT=S.difference_distribution_matrix()
   for a in range(1,256):
      for b in range(1,256):
         if (popcount_py(a)+popcount_py(b)<min and DDT[a][b]>0):
            min=popcount_py(a)+popcount_py(b)
   return min
```

The branching number of this S-Box is 2, which is consistent with that of a pseudo-random S-Box.

5.3 Analysis of the Differential and Linear Spectra

5.3.1 Context

What we want to do here is to compare the differential and linear properties of the CaveTable with those of a pseudo-random S-Box and a structured S-Box which is the Rijindael S-Box (more information in [DR98]). One of the design criteria of the S-Boxes is to have the lowest coefficients possible in the DDT. Else, the system could be vulnerable to differential attacks. We do not need to compare the raw DDT tables, we need to compare the differential spectra. To compare them, we draw on the same graph the scatter plots $k = f(\sigma)$, where σ refers to the DDT (or absolute LAT) values, and k refers to the number of occurrences in the DDT (or absolute LAT). Each scatter plot corresponds to an S-Box spectrum.

The results are in Figure 4.

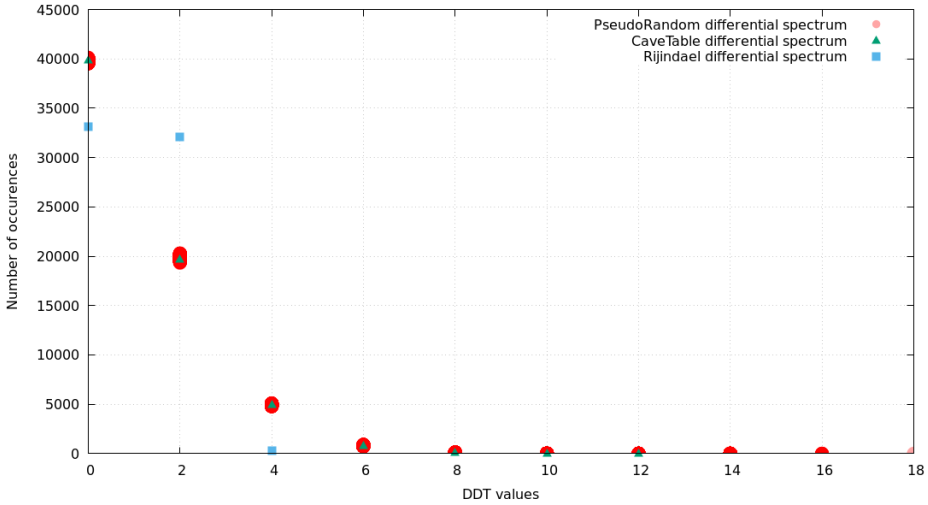


FIGURE 4: Plot representation of the differential spectra of the CaveTable, pseudo-random S-Boxes, and Rijindael

It is worth noticing that even when looking at 1000 pseudo-random S-Boxes, they all have a very similar differential spectrum. As we can see, there is little dispersion in the red scatter plot for a given DDT value. The CaveTable DDT spectrum looks highly consistent with that of a pseudo-random S-Box. In order to translate this similarity in numerical terms, we use the Chi-Squared test.

Definition 1 (Chi-Squared Test)**.** *In order to compare two statistical distributions, we use the Chi-Squared Test. If we want to compare 2 variables O and E, and we have J values for each variable, then the degree of freedom is defined as J-1. We now define*

$$A = \sum_{i=1, E_i \neq 0}^{J} \frac{(O_i - E_i)^2}{E_i} \ .$$

If we take a risk of 0.05, then a table given in Appendix 2.4 gives a critical value A_{crit} such that $P(A \leqslant A_{crit}) = 0.95$. Hence, if $A \leqslant A_{crit}$, we can conclude with a risk of 0.05 that the two distributions are not significantly different.

5.3.2 Differential Spectrum Analysis

We already know that T is an 8×4 S-Box made of sixteen 4×4 permutations. We would like to know if T has similar differential properties as those of an S-Box picked randomly from the set of those with a similar structure.

Definition 2 (Special S-Box)**.** *A special S-Box is an 8×4 S-Box such that all its rows are 4×4 permutations. More precisely, it is a function S such that:*

$$S : \begin{cases} \mathbb{F}_2^4 \times \mathbb{F}_2^4 & \to \mathbb{F}_2^4 \\ (x, y) & \mapsto S_x(y) \ , \end{cases}$$

where S_x is a permutation for all y.

Let us compare the differential spectrum of T with that of pseudo-randomly generated special S-Boxes. The result is plotted in Figure 5. The differential spectrum of

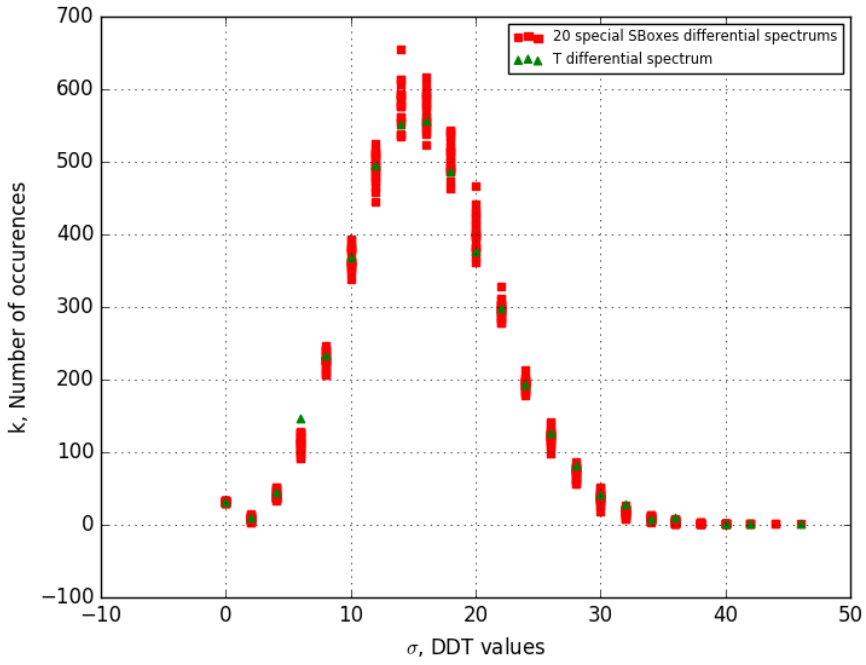


FIGURE 5: Plot representations of T differential spectrum and those of special pseudo-random S-Boxes.

T looks consistent with that of a special S-Box. Indeed, we can use the Chi-squared test to compare the distributions. In order to be statistically more accurate, instead of using 20 pseudo-random special S-Boxes (value chosen for the plot representation), we use 1000. We define $k_\sigma^{\text{pseudorandom}}$ as the average number of occurrences of σ in the DDT of a pseudo-random special S-Box. Similarly, we define k_σ^T as the average number of occurrences of σ in the DDT of T. We decide to ignore the points where $k_\sigma^{\text{pseudorandom}} = 0$. The degree of freedom is equal to 21-1=20. Taking more points into consideration does not affect the Chi-Squared test result. We compute

$$A = \sum_{\sigma, k_\sigma^{\text{pseudorandom}} \neq 0} \frac{(k_\sigma^T - k_\sigma^{\text{pseudorandom}})^2}{k_\sigma^{\text{pseudorandom}}} \ .$$

We get $A = 29.00$. The critical value for a risk of 0.05 is 31.41, and $A \leqslant 31.41$. Hence, the differential spectrum of T is consistent with that of a pseudo-random 8×4 S-Box made of sixteen 4×4 permutations.

Let us now compare the differential spectrum of T with that of a pseudo-random 8×4 S-Box without anymore conditions (rows are not necessarily permutations). It is plotted in Figure 6.

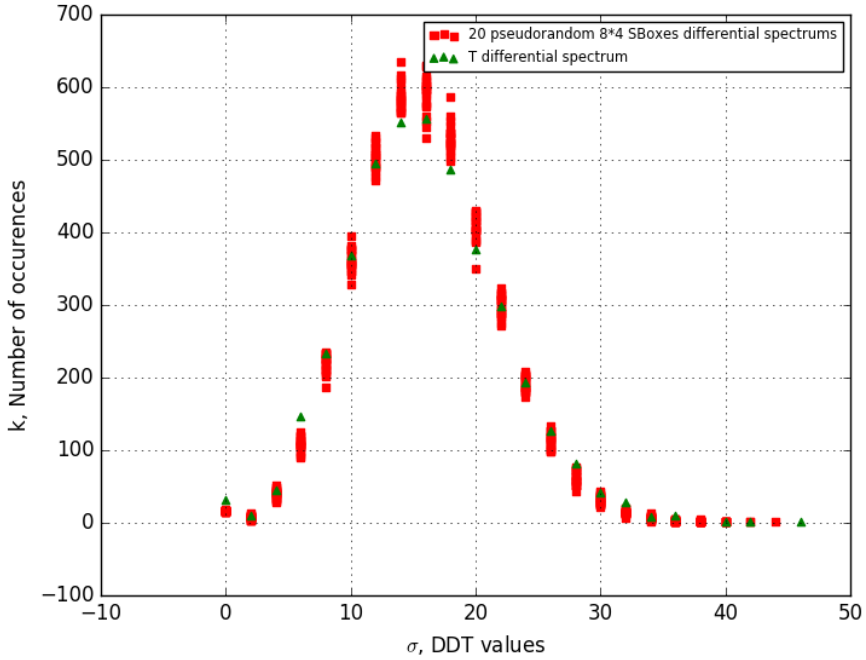


FIGURE 6: Plot representations of T differential spectrum and those of pseudo-random 8×4 S-Boxes

If we do a Chi-Squared test, we get that there are 20 degrees of freedom and A is equal to 90.04. Hence, the Chi-Squared test with a risk of 0.05 fails, it shows that the two distributions are significantly different.

5.3.3 Linear Spectrum Analysis

Let us now take a look at the LAT absolute spectrum (the linear spectrum). The results are presented in Figure 7.

A similar Chi-squared test (with the same risk) shows that T linear spectrum is consistent with that of a pseudo-random 8×4 S-Box made of sixteen 4×4 permutations.

Let us now compare the linear spectrum of T with that of a pseudo-random 8×4 S-Box without any conditions (rows are not necessarily permutations). The result is plotted in Figure 8. There is no need to do a Chi-squared test. Indeed, we can see that one term over two is zero for the T linear spectrum, whereas we have values for each σ from 0 to 45 in the LAT spectrum of the special S-Boxes.

5.3.4 Comparison With Poisson Distribution

In order to model the distribution of a pseudo-random S-Box, a Poisson distribution was suggested in [DR07].

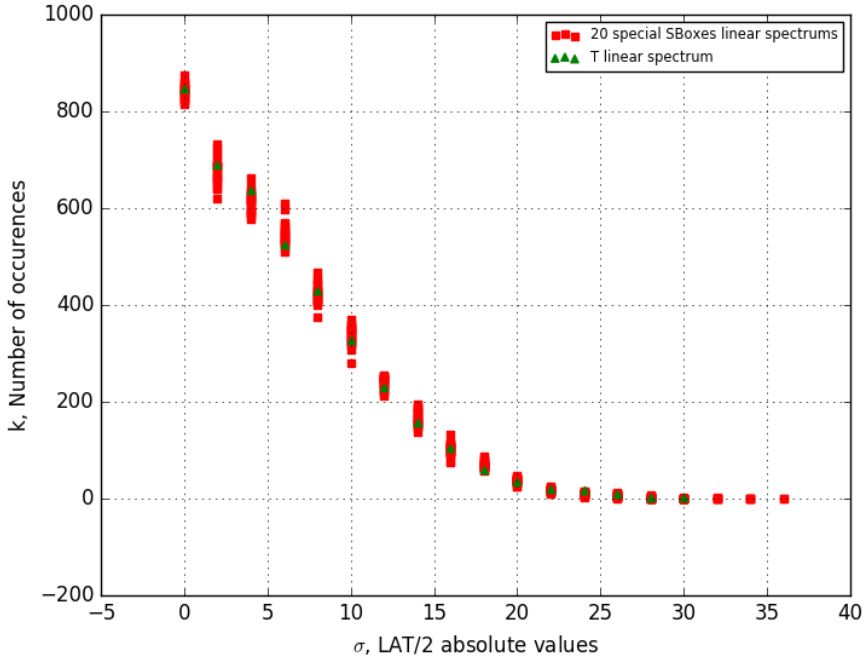


FIGURE 7: Plot representations of T linear spectrum and those of pseudo-random special S-Boxes

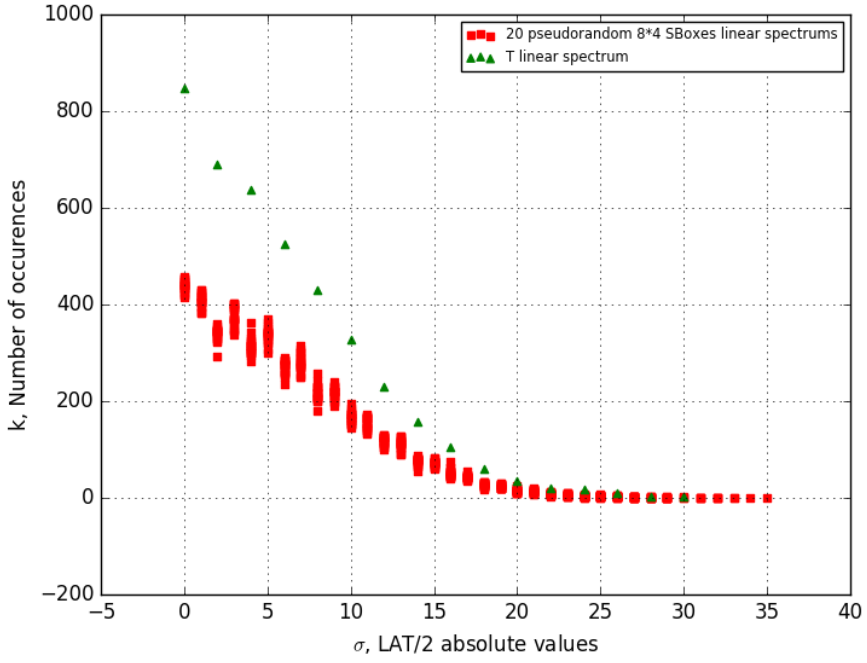


FIGURE 8: Plot representations of T linear spectrum and those of pseudo-random 8×4 S-Boxes

Claim 1. *Let S be a pseudo-random $m \times n$ function such that $m - n$ is small and $m \geq 5$. Then we can make the following approximation:*

$$\mathbb{P}[\mathrm{DDT}[a, b] = 2z] = \mathrm{Poisson}(z, 2^{m-n-1}) = e^{-2^{m-n-1}} \frac{2^{(m-n-1)z}}{z} \ .$$

Even if the Poisson distribution is a good approximation, it seems to not be consistent in our case. Let us compare the differential spectrum of T with the Poisson distribution using the Chi-Squared Test. The expected Poisson spectrum for a 8×4 S-Box would be

$$\{(256 \times 16 \times \mathbb{P}[\mathrm{DDT}[a,b] = 2z], 2z)\} \ .$$

We have a degree of freedom of 24. $A_{crit} = 36.415$ if we take a risk of 0.05. We get $A = 703.91$ which is significantly greater than A_{crit}. If we take a closer look at how A got that high, we see that that there is a great gap regarding the number of zeroes in the DDT. In fact, there are 31 zeroes in T while according to the Poisson distribution, there should be 1.37. This is due to the fact that the approximation is for a pseudo-random S-Box and not a special one. For a special one, some zeroes are already known.

Lemma 2 (Zeroes in a special S-Box)**.** *In a special S-Box, for all $a \in \mathbb{F}_2^4$ such that $a \neq 0$,* $\mathrm{DDT}[(0,a),0] = 0$.

Proof. Let $x = (x_l, x_r) \in \mathbb{F}_2^4 \times \mathbb{F}_2^4$, then

$$\mathrm{DDT}[(0,a),0] = \#\left\{(x_l,x_r) \in \mathbb{F}_2^4 \times \mathbb{F}_2^4 \ , S((x_l,x_r) \oplus (0,a)) \oplus S(x_l,x_r) = 0\right\} \ .$$

However, $S((x_l,x_r) \oplus (0,a)) \oplus S(x_l,x_r) = S(x_l, x_r \oplus a) \oplus S(x_l,x_r)$ and we know that for a special S-Box, the lines are permutations, which means that for $a \neq 0$, $S(x_l, x_r \oplus a) \neq S(x_l,x_r)$, which implies $S((x_l,x_r) \oplus (0,a)) \oplus S(x_l,x_r) \neq 0$. Hence, $\mathrm{DDT}[(0,a),0] = 0$ □

The first line of the DDT is composed of one 256 and then zeroes.

Lemma 3 (Zeroes in the first line of the DDT)**.** *Let S be a 8×4 S-Box, $\forall b \neq 0$,* $\mathrm{DDT}[0,b] = 0$, *and* $\mathrm{DDT}[0,0] = 2^n$.

Proof. Using the definition of the DDT,

$$\mathrm{DDT}[0,b] = \#\left\{x, S(x) \oplus S(x) = b\right\} \ .$$

We have that $S(x) \oplus S(x) = 0$. Hence, $\mathrm{DDT}[0,0] = 2^n$ since all x in $\mathbb{F}_2^n$ satisfy the relation $S(x) \oplus S(x) = 0$. We also get that for $b \neq 0$, $\mathrm{DDT}[0,b] = 0$. □

Let us do another Chi-Squared test disregarding the number of zeroes. To do that we also twist a little bit the Poisson distribution, instead of multiplying the probability to get a zero by 256×16 (size of the DDT), we multiply it by 255×16 (we ignore the first line). As for the table T, we do not consider the number of zeroes in the DDT. We have a degree of freedom of 23, so $A_{crit} = 35.17$ with the same risk of 0.05. We get $A = 65.22$, meaning that $A \geq A_{crit}$. Hence, we exclude the hypothesis that the DDT spectrum of T follows a Poisson distribution. There might be two underlying reasons for that. First, the approximation given is for a general pseudo-random function and there is no approximation for a special S-Box. Second, in order to do such an approximation, the authors made the assumption that all DDT coefficients are independent, which is technically not true since the sum of the coefficients in a line of the DDT is always equal to 2^n. These results lead me to propose the following open problem.

Open Problem 1. *Let S be a pseudo-random $n \times m$ S-Box. Find a DDT spectrum approximation that is more accurate than the Poisson approximation.*

I think that such an approximation could be found by looking at the partitions of 2^n into 2^m integers and more specifically, looking at the statistical distribution for, let's say, the first integer of the 2^m integers of a partition. However, such partitions are way to numerous to do such a statistical analysis directly with a computer.

6 Relations With Cryptographic Algorithms

6.1 CMEA

CMEA (Cellular Message Encryption Algorithm) is a block cipher that is described in [Qui09]. The CaveTable is involved in the process of making the T-Box as follows:

$$\begin{aligned} I(z) &= C((x \oplus k_0) + k_1) + x \ , \\ J(z) &= C((I(z) \oplus k_2) + k_3) + x \ , \\ K(z) &= C((J(z) \oplus k_4) + k_5) + x \ , \\ \text{Tbox}(z) &= C((K(z) \oplus k_6) + k_7) + x \ , \end{aligned}$$

with '+' being the addition modulo 256. k_0,...k_7 are 8 bytes derived from a 64-bit key. This information may not be helpful when it comes to analysing the CaveTable. However, we note that another S-Box is used in the process of making the T-Box. This process is called the "Jumble Process" and may be found in [III91]. We can then make the hypothesis that the CaveTable is not that important for CMEA, rather it is important for the CAVE algorithm. CMEA is a weak block cipher as many attacks have been published, like in [WSK97]. The attacks use the fact that the CaveTable has a skewed distribution and the T-Box too. Indeed, the CaveTable is not a permutation, some values appear up to 4 times, 92 values are absent. Such a statistical bias helped attackers to lower the number of cases they had to consider. In fact, the main weakness in the T-Box is that if we want to know Tbox(z), we know that Tbox(z) $- z$ is in the look-up table of the CaveTable, we can hence rule out 92 possibilities, which considerably simplifies the attack in term of complexity.

6.2 CAVE

The CaveTable is not only used in CMEA. In fact, it is also used in the CAVE algorithm with is presented in [Qui09]. The S-Box is split into 2 tables, table0 and table1 which are respectively the low-weight 4 bits and the high-weight 4 bits. table0 is used to pseudo-randomly mix registers. Hence, the fact that the rows of T are permutations is not a coincidence, for if it was not a permutation, it could not be used to fully mix the registers. We however note that the design criteria of these permutations is not given, rather we understand how it is used. The fact that the high-order 4 bits are permutations is not explained either. Hence, it is possible that these permutations were generated pseudo-randomly as they were merely used to mix registers. The corresponding CaveTable may have also been used for CMEA as they were low on space. However, this is only a conjecture.

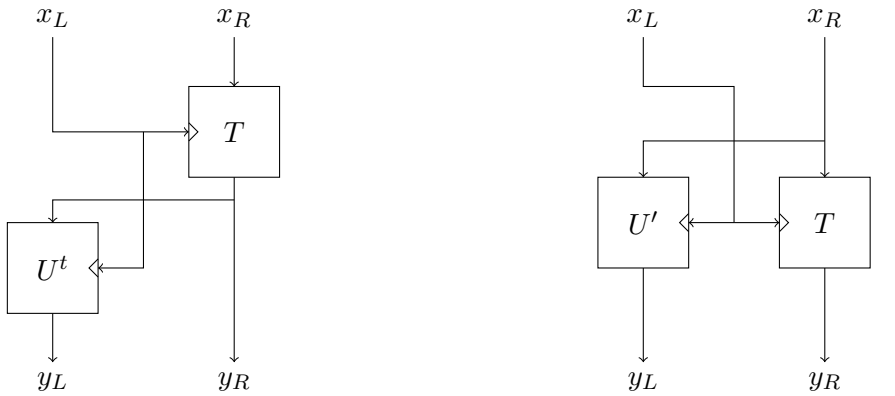
7 Possible Improvements

The CaveTable may have been designed for CAVE and had to be reused for CMEA for memory storage issues. We know that the low-weight 4 bits of the CaveTable had to form a permutation for a given row. However, nothing in the CAVE algorithm

suggests that the high-weight 4 bits had to be a permutation for a given row. Other S-Boxes fulfilling the given criteria exist and are more resistant to attacks that the given CaveTable. For instance, using a 3-round Feistel Network or a 2-round SPN results in getting an 8×8 permutation S-Box. Both satisfy CAVE criteria and are easily physically implemented. Some people suggested using the AES S-Box for CMEA. However, it does not fulfill the criteria of CAVE as the 4 low-weight bits are not a permutation for each row. Using a permutation S-Box results in cancelling the statistical bias of the CaveTable that was used for attacks.

8 Conclusion

We exhibited two possible representations of the CaveTable. We now know that the one that was used for designing the S-Box was the parallel representation given in Figure 9b, with T corresponding to table0 and U' corresponding to table1, as defined in CAVE. We understand why the rows of T are permutation, this is because of



(A) Serial representation. (B) Parallel representation.

FIGURE 9: First decompositions of CMEA S-Box.

requirements related to CAVE. However, we do not know why the same property is found in U', even though we suspect that the same process for generating T was used to generate U' so that a full 8×8 S-Box was made. All our research shows that it is very likely that T is an 8×4 S-Box constructed using sixteen pseudo-random permutations. The CaveTable fulfills the CAVE requirements, however, that which was relevant for CAVE was not for CMEA: such a design opened the gates for many attacks. It is very likely that the same S-Box was used again for CMEA as designers were low on space.

Acknowledgements

The report Latex style was originally created by Steve R. Gunn and modified into a template by Sunil Patel. This second major version of this template was made by Vel. The template was downloaded from http://www.LaTeXTemplates.com under the license CC BY-NC-SA 3.0 (http://creativecommons.org/licenses/by-nc-sa/3.0/). Some changes were made on it. I want to thank my internship supervisor for teaching and helping me during the internship, and also for reviewing this report. I thank the administrative staff for assisting me during my administrative procedures. I thank the project team SECRET for the help they provided and for helping me getting integrated during the internship. I also want to thank the Sage, LIGHTER and PEIGEN developers as these tools were really useful for this research. Finally, I thank my family for their encouragement and moral support during this Summer internship.

Appendices

1 Pseudo Random Permutations Properties

```
from random import shuffle

if __name__ == '__main__':
   res=""
   for i in range(100):
      s=range(16)
      shuffle(s)
      resi=""
      for i in range(16):
         resi+=hex(s[i])[2]
      res+=resi
   print(res)
```

and the corresponding bash script:

```
#!/bin/bash

s=$(python pseudo-random.py)

for i in {0..1584..16}; do
    timeout 10m ./non-lin-search -a -v -r 5 -o "${s:i:16}"
done
```

2 Dictionary Generation

2.1 Components Classification

```
import sage.all
from sage.crypto.mq import *
from sboxU import *

#SBox CMEA
S = SBox([
    0xd9,0x23,0x5f,0xe6,0xca,0x68,0x97,0xb0,0x7b,0xf2,0x0c,0x34,0x11,0xa5,0x8d,0x4e,
    0x0a,0x46,0x77,0x8d,0x10,0x9f,0x5e,0x62,0xf1,0x34,0xec,0xa5,0xc9,0xb3,0xd8,0x2b,
    0x59,0x47,0xe3,0xd2,0xff,0xae,0x64,0xca,0x15,0x8b,0x7d,0x38,0x21,0xbc,0x96,0x00,
    0x49,0x56,0x23,0x15,0x97,0xe4,0xcb,0x6f,0xf2,0x70,0x3c,0x88,0xba,0xd1,0x0d,0xae,
    0xe2,0x38,0xba,0x44,0x9f,0x83,0x5d,0x1c,0xde,0xab,0xc7,0x65,0xf1,0x76,0x09,0x20,
    0x86,0xbd,0x0a,0xf1,0x3c,0xa7,0x29,0x93,0xcb,0x45,0x5f,0xe8,0x10,0x74,0x62,0xde,
    0xb8,0x77,0x80,0xd1,0x12,0x26,0xac,0x6d,0xe9,0xcf,0xf3,0x54,0x3a,0x0b,0x95,0x4e,
    0xb1,0x30,0xa4,0x96,0xf8,0x57,0x49,0x8e,0x05,0x1f,0x62,0x7c,0xc3,0x2b,0xda,0xed,
    0xbb,0x86,0x0d,0x7a,0x97,0x13,0x6c,0x4e,0x51,0x30,0xe5,0xf2,0x2f,0xd8,0xc4,0xa9,
    0x91,0x76,0xf0,0x17,0x43,0x38,0x29,0x84,0xa2,0xdb,0xef,0x65,0x5e,0xca,0x0d,0xbc,
    0xe7,0xfa,0xd8,0x81,0x6f,0x00,0x14,0x42,0x25,0x7c,0x5d,0xc9,0x9e,0xb6,0x33,0xab,
```

```
    0x5a,0x6f,0x9b,0xd9,0xfe,0x71,0x44,0xc5,0x37,0xa2,0x88,0x2d,0x00,0xb6,0x13,0xec,
    0x4e,0x96,0xa8,0x5a,0xb5,0xd7,0xc3,0x8d,0x3f,0xf2,0xec,0x04,0x60,0x71,0x1b,0x29,
    0x04,0x79,0xe3,0xc7,0x1b,0x66,0x81,0x4a,0x25,0x9d,0xdc,0x5f,0x3e,0xb0,0xf8,0xa2,
    0x91,0x34,0xf6,0x5c,0x67,0x89,0x73,0x05,0x22,0xaa,0xcb,0xee,0xbf,0x18,0xd0,0x4d,
    0xf5,0x36,0xae,0x01,0x2f,0x94,0xc3,0x49,0x8b,0xbd,0x58,0x12,0xe0,0x77,0x6c,0xda])

def spectre_dico(mat):
   d=defaultdict(int)
   for a in range(0,mat.nrows()):
      for b in range(0,mat.ncols()):
         d[mat[a][b]]+=1
   return d

def spectrum_to_int(s): #if we just want to classify by the DDT or the absolute LAT
   r=0
   for k in range(0,16,2):
      r+=s[k]<<(4*k)
   return r

def intLAT_and_DDT(ddtint,latint):#if we want to classify by both the DDT
   r=0
   r=(ddtint<<68)+latint
   return r

dico_DDT=defaultdict(list) #as we first just want to classify by the DDT

#We store the DDTs and the LATs in corresponding lists
for j in range(16):
   DDTTi.append(TiSBox[j].difference_distribution_matrix())
   DDTUi.append(UiSBox[j].difference_distribution_matrix())
   DDTUtransposei.append(UtransposeiSBox[j].difference_distribution_matrix())
   LATTi.append(TiSBox[j].linear_approximation_matrix())
   LATUi.append(UiSBox[j].linear_approximation_matrix())
   LATUtransposei.append(UtransposeiSBox[j].linear_approximation_matrix()))

for i in range(16):
   dico_DDT[spectrum_to_int(spectre_dico(DDTTi[i]))].append("T"+str(i))
   dico_DDT[spectrum_to_int(spectre_dico(DDTUi[i]))].append("U"+str(i))
   dico_DDT[spectrum_to_int(spectre_dico(DDTUtransposei[i]))].append("Utranspose"+str(i))
```

2.2 $T[i+k_1]+k_0$ classification

```
import itertools

def absol(mat):
       return [[ abs(mat[i][j]) for j in range (mat.ncols())] for i in
           range(mat.nrows())]

dico_DDT_and_LAT=defaultdict(list)
for i,k1,k0 in itertools.product(range(16),range(16),range(16)):
   signature=(i,k1,k0) #used to identify the component we are looking at

   latint=spectrum_to_int(spectre_dico(matrix(absol(SBox([(T[i][(x+k1)%16]+k0)%16
   for x in range(16)]).linear_approximation_matrix()))))

   ddtint=spectrum_to_int(spectre_dico(SBox([(T[i][(x+k1)%16]+k0)%16
   for x in range(16)]).difference_distribution_matrix()))

   dico_DDT_and_LAT[intLAT_et_DDT(ddtint,latint)].append(signature)
print(dict(dico_DDT_and_LAT))
```

2.3 Affine Equivalence

```
for key in dico_DDT_and_LAT:
    for (i,k1,k0) in dico_DDT_and_LAT[key]:
        for (j,k1j,k0j) in dico_DDT_and_LAT[key]:
            if((i,k1,k0)!=(j,k1j,k0j)):
                if(len(affine_equivalence([int(x) for x in [(T[i][(x+k1)%16]+k0)%16
                    for x in range(16)]] ,
                                          [int(x) for x in [(T[j][(x+k1j)%16]+k0j)%16
                                              for x in range(16)]]))!=0):
                    print("i="+str(i)+" k1="+str(k1)+" k0=" + str(k0)+" j= "+str(j)+"
                        k1j="+str(k1j)+" k0j="+str(k0j))
                    print(affine_equivalence([int(x) for x in
                        [(T[i][(x+k1)%16]+k0)%16 for x in range(16)]] ,[int(x) for x
                        in [ (T[j][(x+k1j)%16]+k0j)%16 for x in range(16)]]))
```

2.4 Chi-Squared Table

	Risk								
d.f.	**.995**	**.99**	**.975**	**.95**	**.9**	**.1**	**.05**	**.025**	**.01**
1	0.00	0.00	0.00	0.00	0.02	2.71	3.84	5.02	6.63
2	0.01	0.02	0.05	0.10	0.21	4.61	5.99	7.38	9.21
3	0.07	0.11	0.22	0.35	0.58	6.25	7.81	9.35	11.34
4	0.21	0.30	0.48	0.71	1.06	7.78	9.49	11.14	13.28
5	0.41	0.55	0.83	1.15	1.61	9.24	11.07	12.83	15.09
6	0.68	0.87	1.24	1.64	2.20	10.64	12.59	14.45	16.81
7	0.99	1.24	1.69	2.17	2.83	12.02	14.07	16.01	18.48
8	1.34	1.65	2.18	2.73	3.49	13.36	15.51	17.53	20.09
9	1.73	2.09	2.70	3.33	4.17	14.68	16.92	19.02	21.67
10	2.16	2.56	3.25	3.94	4.87	15.99	18.31	20.48	23.21
11	2.60	3.05	3.82	4.57	5.58	17.28	19.68	21.92	24.72
12	3.07	3.57	4.40	5.23	6.30	18.55	21.03	23.34	26.22
13	3.57	4.11	5.01	5.89	7.04	19.81	22.36	24.74	27.69
14	4.07	4.66	5.63	6.57	7.79	21.06	23.68	26.12	29.14
15	4.60	5.23	6.26	7.26	8.55	22.31	25.00	27.49	30.58
16	5.14	5.81	6.91	7.96	9.31	23.54	26.30	28.85	32.00
17	5.70	6.41	7.56	8.67	10.09	24.77	27.59	30.19	33.41
18	6.26	7.01	8.23	9.39	10.86	25.99	28.87	31.53	34.81
19	6.84	7.63	8.91	10.12	11.65	27.20	30.14	32.85	36.19
20	7.43	8.26	9.59	10.85	12.44	28.41	31.41	34.17	37.57
22	8.64	9.54	10.98	12.34	14.04	30.81	33.92	36.78	40.29
24	9.89	10.86	12.40	13.85	15.66	33.20	36.42	39.36	42.98
26	11.16	12.20	13.84	15.38	17.29	35.56	38.89	41.92	45.64
28	12.46	13.56	15.31	16.93	18.94	37.92	41.34	44.46	48.28
30	13.79	14.95	16.79	18.49	20.60	40.26	43.77	46.98	50.89
32	15.13	16.36	18.29	20.07	22.27	42.58	46.19	49.48	53.49
34	16.50	17.79	19.81	21.66	23.95	44.90	48.60	51.97	56.06
38	19.29	20.69	22.88	24.88	27.34	49.51	53.38	56.90	61.16
42	22.14	23.65	26.00	28.14	30.77	54.09	58.12	61.78	66.21
46	25.04	26.66	29.16	31.44	34.22	58.64	62.83	66.62	71.20
50	27.99	29.71	32.36	34.76	37.69	63.17	67.50	71.42	76.15
55	31.73	33.57	36.40	38.96	42.06	68.80	73.31	77.38	82.29
60	35.53	37.48	40.48	43.19	46.46	74.40	79.08	83.30	88.38
65	39.38	41.44	44.60	47.45	50.88	79.97	84.82	89.18	94.42
70	43.28	45.44	48.76	51.74	55.33	85.53	90.53	95.02	100.43
75	47.21	49.48	52.94	56.05	59.79	91.06	96.22	100.84	106.39
80	51.17	53.54	57.15	60.39	64.28	96.58	101.88	106.63	112.33
85	55.17	57.63	61.39	64.75	68.78	102.08	107.52	112.39	118.24
90	59.20	61.75	65.65	69.13	73.29	107.57	113.15	118.14	124.12
95	63.25	65.90	69.92	73.52	77.82	113.04	118.75	123.86	129.97
100	67.33	70.06	74.22	77.93	82.36	118.50	124.34	129.56	135.81

TABLE 5: Chi-squared Distribution Table (d.f. means "degree of freedom").

Bibliography

[B92] Rev B. TIA IS-54 Appendix A, "Dual-mode Cellular System: Authentication, Message Encryption, Voice Privacy Mask Generation, Shared Secret Data Generation, A-Key Verification, and Test Data", Feb 1992.

[BGLS19] Zhenzhen Bao, Jian Guo, San Ling, and Yu Sasaki. Sok: Peigen – a platform for evaluation, implementation, and generation of s-boxes. Cryptology ePrint Archive, Report 2019/209, 2019. https://eprint.iacr.org/2019/209.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology-CRYPTO' 90*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[Dev17] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 7.5.1)*, 2017. http://www.sagemath.org.

[DR98] Joan Daemen and Vincent Rijmen. AES submission document on Rijndael. Available online at https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf#page=1., June 1998.

[DR07] Joan Daemen and Vincent Rijmen. Probability distributions of correlation and differentials in block ciphers. *Journal of Mathematical Cryptology*, 1(3):221–242, 2007.

[III91] J.A Reeds III. Cryptosystem for cellular telephony, U.S Patent 5,159,634, Sept. 1991.

[JPST17] Jérémy Jean, Thomas Peyrin, Siang Sim, and Jade Tourteaux. Optimizing implementations of lightweight building blocks. *IACR Transactions on Symmetric Cryptology*, 2017(4):130–168, Dec. 2017.

[MY93] Mitsuru Matsui and Atsuhiro Yamagishi. A new method for known plaintext attack of feal cipher. In Rainer A. Rueppel, editor, *Advances in Cryptology — EUROCRYPT' 92*, volume 1440 of *Lecture Notes in Computer Science*, pages 81–91, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[Per17] Léo Perrin. *Cryptanalysis, Reverse-Engineering and Design of Symmetric Cryptographic Algorithms*. PhD thesis, University of Luxembourg, Belval, Luxembourg, 2017.

[Per19] Léo Perrin. SboxU (S-box utils). Available online at https://github.com/lpp-crypto/sboxU, 2019.

[Qui09] Frank Quick. Common cryptographic algorithms. Published by the 3rd Generation Partnership Project 2 "3GPP2". Available online at https://www.3gpp2.org/Public_html/Specs/S.S0053-0_v2.0.pdf., 2009.

[TCG92] Anne Tardy-Corfdir and Henri Gilbert. A known plaintext attack of feal-4 and feal-6. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 172–182, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[WSK97] David Wagner, Bruce Schneier, and John Kelsey. Cryptanalysis of the cellular message encryption algorithm. In Burton S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 526–537, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.